ShopAppstoreLib - PHP Documentation

Release 0.2

DreamCommerce SA

March 04, 2016

Contents

This is an API/functional documentation of shop-appstore-lib. Here, you'll find an useful information on installation and usage of this library.

Contents:

CHAPTER 1

Library

In order to create a shop application in easy way, we created a SDK, which provides functions allowing payment handling and communications with resources.

1.1 Configuration

Library needs these parameters to be configured:

- Shop URL - each REST-ful API request needs to specify an entry point - a shop URL
- Application ID
- Secret

The configuration is done by specifying values in client object factory, eg:

```
$client = Client::factory(
    Client::ADAPTER_OAUTH,
    array(
        'entrypoint'=>$params['shop_url'],
        'client_id'=>$app['app_id'],
        'client_secret'=>$app['app_secret']
    )
);
```

1.2 Debugging

SDK raises errors using exceptions. Explore library exceptions: *Classes list*.

1.2.1 Debug mode

SDK allows to activate debug mode. Its purpose is to log all requests, responses and headers.

The debugging may be enabled by defining `DREAMCOMMERCE_DEBUG` constant.

value	meaning
false	debug mode is disabled
true	debug mode is enabled

You can define the constant in your source code:

```
define('DREAMCOMMERCE_DEBUG', true);
```

1.2.2 Logging messages

SDK allows to log all messages to stream.

The log file may be set by defining `DREAMCOMMERCE_LOG_FILE` constant.

value	meaning
false	logging is disabled - except errors but you still need to specify log file
string	file path or stream (i.e. `"php://stdout"`, `"logs/application.log"`)

You can define the constant in your source code:

```
define('DREAMCOMMERCE_LOG_FILE', "php://stdout");
```

Messages can be passed to simple logger class using multiple priorities:

```
\DreamCommerce\ShopAppstoreLib\Logger::debug("debug message");
\DreamCommerce\ShopAppstoreLib\Logger::info("informational message");
\DreamCommerce\ShopAppstoreLib\Logger::notice("notice message");
\DreamCommerce\ShopAppstoreLib\Logger::warning("warning message");
\DreamCommerce\ShopAppstoreLib\Logger::error("error message");
\DreamCommerce\ShopAppstoreLib\Logger::critical("critical message");
\DreamCommerce\ShopAppstoreLib\Logger::alert("alert message");
\DreamCommerce\ShopAppstoreLib\Logger::emergency("emergency message");
```

Debug messages are filtered if debug mode is disabled.

1.2.3 Own logger

If you prefer to take over all logging, simply create your own class implementing PSR-3 Logger Interface and pass it to the library.

```
$client = Client::factory(
    Client::ADAPTER_OAUTH,
    array(
        'entrypoint'=>$shop->getShopUrl(),
        'client_id'=>$this->getAppId(),
        'client_secret'=>$this->getAppSecret()
    )
);

$client->setAccessToken($tokens->getAccessToken());

$logger = new MyLogger();
$client->setLogger($logger);
```

Note: If you use your own logger, all constants described above are ignored.

1.2.4 Catching exceptions

A code example using exceptions handling:

```
try{
    $client = Client::factory(
        Client::ADAPTER_OAUTH,
        array(
            'entrypoint'=>'https://example.com',
            'client_id'=>'app_id',
            'client_secret'=>'app_secret'
        )
    );

    $client->setAccessToken('TOKEN');

    // fetch collection/object
    $product = new \DreamCommerce\ShopAppstoreLib\Resource\Product($client);
    $list = $product->get();

    foreach($list as $item){
        //...
    }

} catch (\DreamCommerce\ShopAppstoreLib\Resource\Exception\NotFoundException $ex) {
    \DreamCommerce\ShopAppstoreLib\Logger::debug('resource not found', array((string)$ex));
} catch (\DreamCommerce\ShopAppstoreLib\Exception\ResourceException $ex) {
    // resource error
    \DreamCommerce\ShopAppstoreLib\Logger::error($ex);
}
```

Using default logger library, all traffic is being logged unless you disable debug mode. More over, if debugging is disabled, logger will catch all exceptions that are not covered by particular ones. This means if server returns HTTP 500, this exception data will be stored. You can disable it at all by not setting `DREAMCOMMERCE_LOG_FILE`.

If you need to take more control on data logging, implement your own logger.

Each exception lets to access an exception of lower layer, eg. HTTP response. Simply use standard exception's method `getPrevious` on every exception.

```
try{

    // ...

} catch (\DreamCommerce\ShopAppstoreLib\Client\Exception\Exception $ex) {
    \DreamCommerce\Logger::error(sprintf("Client error: %s", $ex->getMessage()));

    $prev = $ex->getPrevious();

    if($prev instanceof \DreamCommerce\ShopAppstoreLib\Exception\HttpException){
        \DreamCommerce\ShopAppstoreLib\Logger::error(sprintf("HTTP error: %s", $prev->getMessage()));

        if($prev->getCode() == \DreamCommerce\ShopAppstoreLib\Exception\HttpException::QUOTA_EXCEEDED
            \DreamCommerce\ShopAppstoreLib\Logger::warning("Quota exceeded");
        }
    }

} catch (\DreamCommerce\ShopAppstoreLib\Exception\ResourceException $ex) {
    \DreamCommerce\ShopAppstoreLib\Logger::error(sprintf("Resource error: %s", $ex->getMessage()));
}
```

1.3 Examples

1.3.1 Configuration

```
return array(
    /*
     * Application ID generated by AppStore
     */
    'appId' => '',

    /*
     * Secret generated by AppStore
     */
    'appSecret' => '',

    /*
     * AppStore Secret generated by AppStore
     */
    'appstoreSecret' => '',

    /*
     * Enable debug mode or not
     */
    'debug' => false,

    /*
     * Path to log file or empty to disable logging
     */
    'logFile' => "logs/application.log",


    /*
     * timezone of the application
     *
     * Value is passed to date_default_timezone_set function
     */
    'timezone' => 'Europe/Warsaw',

    'php' => array(
        /*
         * This determines whether errors should be printed to the screen as
         * part of the output or if they should be hidden from the user
         *
         * Value is passed to ini_set function
         */
        'display_errors' => 'off'
    )
);
```

1.3.2 Bootstrapping

```
// force utf-8 as primary encoding
if (PHP_VERSION_ID < 50600) {
    mb_internal_encoding('utf-8');
} else {
```

```
    ini_set('default_charset', 'utf-8');
}

// internal autoloader
spl_autoload_register(function($class){
    $class = str_replace('\\', '/', $class);
    require 'src/'.$class.'.php';
});

// composer autoloader - you can enable it on by uncommenting this line:
//require 'vendor/autoload.php';

// use config from previous example
$config = require __DIR__. '/Config.php';

// various PHP configuration values
date_default_timezone_set($config['timezone']);
ini_set('display_errors', $config['php']['display_errors']);

// check debug mode options
$debug = false;
if(isset($config['debug'])){
    if($config['debug']){
        $debug = true;
    }
}
// check environment variable
if(getenv('DREAMCOMMERCE_DEBUG')){
    $debug = true;
}
define("DREAMCOMMERCE_DEBUG", $debug);

// log errors to stdout by default
$logFile = "php://stdout";
if(isset($config['logFile'])){
    if($config['logFile']){
        $logFile = $config['logFile'];
    }else{
        $config['logFile'] = false;
    }
}
define("DREAMCOMMERCE_LOG_FILE", $logFile);

return $config;
```

1.3.3 REST GET

```
use DreamCommerce\ShopAppstoreLib\Client;
use DreamCommerce\ShopAppstoreLib\Exception\ClientException;
use DreamCommerce\ShopAppstoreLib\Exception\ResourceException;

$config = require 'bootstrap.php';

try {
    // set custom retries count
    // it will throw HttpException if the limit is too low
```

```
    \DreamCommerce\ShopAppstoreLib\Http::setRetryLimit(2);

    $client = Client::factory(
        Client::ADAPTER_OAUTH,
        array(
            'entrypoint'=>'https://myshop.example.com',
            'client_id'=>$config['appId'],
            'client_secret'=>$config['appSecret']
        )
    );

    $client->setAccessToken('INSERT TOKEN HERE');

    $resource = new \DreamCommerce\ShopAppstoreLib\Resource\Product($client);
    // or
    $resource = $client->products;

    // particular object, with ID=1
    $result = $resource->get(1);

    // list of objects
    $result = $resource->get();

    // list of objects (page 3) with filtering/limiting:
    $result = $resource->filters(array('translations.name' => array('=', 'laptop')))->page(3)->limit

    printf("Found: %d\n", $result->count);
    printf("Page: %d of %d\n", $result->page, $result->pages);
    printf("Iterating over products:\n");
    foreach ($result as $i) {
        printf("ID #%d\n", $i->product_id);
        // or - for your convenience:
        //printf("ID #%d\n", $i['product_id']);
    }
} catch (ClientException $ex) {
    $client->getLogger()->error("An error occurred during the Client initialization: ".Client::getEr
} catch (ResourceException $ex) {
    $client->getLogger()->error("An error occurred during Resource access: ".Client::getError($ex));
}
```

1.3.4 REST POST

```
use DreamCommerce\ShopAppstoreLib\Client;
use DreamCommerce\ShopAppstoreLib\Exception\ClientException;
use DreamCommerce\ShopAppstoreLib\Exception\ResourceException;

$config = require 'bootstrap.php';

try {
    // set custom retries count
    // it will throw HttpException if the limit is too low
    \DreamCommerce\ShopAppstoreLib\Http::setRetryLimit(2);

    $client = Client::factory(
        Client::ADAPTER_OAUTH,
        array(
            'entrypoint'=>'https://myshop.example.com',
```

```
            'client_id'=>$config['appId'],
            'client_secret'=>$config['appSecret']
        )
    );

    $client->setAccessToken('INSERT TOKEN HERE');

    $resource = new \DreamCommerce\ShopAppstoreLib\Resource\Producer($client);
    // or
    $resource = $client->producers;

    $insertedId = $resource->post(array(
        'name' => 'Awesome Manufacturer!',
        'web' => 'http://example.org'
    ));

    // or:
    $data = new stdClass();
    $data->name = 'Awesome Manufacturer!';
    $data->web = 'http://example.org';
    $insertedId = $resource->post($data);


} catch (ClientException $ex) {
    $client->getLogger()->error("An error occurred during the Client initialization: ".Client::getEr
} catch (ResourceException $ex) {
    $client->getLogger()->error("An error occurred during Resource access: ".Client::getError($ex));
}
```

1.3.5 REST PUT

```
use DreamCommerce\ShopAppstoreLib\Client;
use DreamCommerce\ShopAppstoreLib\Exception\ClientException;
use DreamCommerce\ShopAppstoreLib\Exception\ResourceException;

$config = require 'bootstrap.php';

try {
    // set custom retries count
    // it will throw HttpException if the limit is too low
    \DreamCommerce\ShopAppstoreLib\Http::setRetryLimit(2);

    $client = Client::factory(
        Client::ADAPTER_OAUTH,
        array(
            'entrypoint'=>'https://myshop.example.com',
            'client_id'=>$config['appId'],
            'client_secret'=>$config['appSecret']
        )
    );

    $client->setAccessToken('INSERT TOKEN HERE');

    $resource = new \DreamCommerce\ShopAppstoreLib\Resource\Producer($client);
    // or
    $resource = $client->producers;
```

```
    $insertedId = $resource->put(2, array(
        'name' => 'Awesome Manufacturer!'
    ));

    $client->getLogger()->info("Object modified");

} catch (ClientException $ex) {
    $client->getLogger()->error("An error occurred during the Client initialization: ".Client::getEr
} catch (ResourceException $ex) {
    $client->getLogger()->error("An error occurred during Resource access: ".Client::getError($ex));
}
```

1.3.6 REST DELETE

```
use DreamCommerce\ShopAppstoreLib\Client;
use DreamCommerce\ShopAppstoreLib\Exception\ClientException;
use DreamCommerce\ShopAppstoreLib\Exception\ResourceException;

$config = require 'bootstrap.php';

try {
    // set custom retries count
    // it will throw HttpException if the limit is too low
    \DreamCommerce\ShopAppstoreLib\Http::setRetryLimit(2);

    $client = Client::factory(
        Client::ADAPTER_OAUTH,
        array(
            'entrypoint'=>'https://myshop.example.com',
            'client_id'=>$config['appId'],
            'client_secret'=>$config['appSecret']
        )
    );

    $client->setAccessToken('INSERT TOKEN HERE');

    $resource = new \DreamCommerce\ShopAppstoreLib\Resource\Producer($client);
    // or
    $resource = $client->producers;

    $result = $resource->delete(41);
    $client->getLogger()->info("An object was successfully deleted");

} catch (ClientException $ex) {
    $client->getLogger()->error("An error occurred during the Client initialization: ".Client::getEr
} catch (ResourceException $ex) {
    $client->getLogger()->error("An error occurred during Resource access: ".Client::getError($ex));
}
```

1.3.7 Token refreshing

```
use DreamCommerce\ShopAppstoreLib\Client;
use DreamCommerce\ShopAppstoreLib\Exception\ClientException;

$config = require 'bootstrap.php';
```

```
try {

    $client = Client::factory(
        Client::ADAPTER_OAUTH,
        array(
            'entrypoint'=>'https://myshop.example.com',
            'client_id'=>$config['appId'],
            'client_secret'=>$config['appSecret']
        ]
    );

    $client->setAccessToken('INSERT TOKEN HERE');

    $client->getLogger()->info("Token has been successfully refreshed");
} catch (ClientException $ex) {
    $client->getLogger()->error("An error occurred during the Client request: ".$ex->getMessage());
}
```

1.4 Installation

1.4.1 requirements

This library requires following aspects of interpreter to be satisfied:

- PHP 5.3 or higher
- SSL support enabled
- `allow_url_fopen` flag set to `on`
- compiled JSON extension (in the most of cases, available out-of-the-box)

1.4.2 installation

There are two ways to get and install the SDK into application:

- using Composer - in project directory call `composer require dreamcommerce/shop-appstore-lib`
- manual archive extraction - download an archive and extract its contents into project directory

1.5 Typical tasks

1.5.1 Handling billing system events

```
$handler = $this->handler = new Handler(
    'https://myshop.example.com', 'application ID', 'Secret', 'AppStore Secret'
);

$handler->subscribe('install', 'installHandler');   // function name
// $handler->subscribe('install', $installHandler);   // lambda
// $handler->subscribe('install', array($this, 'installHandler'));   // object method

//$handler->subscribe('billing_install', array($this, 'billingInstallHandler'));
```

```
//$handler->subscribe('billing_subscription', array($this, 'billingSubscriptionHandler'));
//$handler->subscribe('uninstall', array($this, 'uninstallHandler'));
```

Passed callback will be executed as an action handler.

1.5.2 Getting OAuth token

Token exchange is performed with the authorization key got during the application install.

The most convenient way is to exchange this code during the install. In billing system entry point it's enough to use:

```
$client = Client::factory(
    Client::ADAPTER_OAUTH,
    array(
        'entrypoint'=>'https://shop.url',
        'client_id'=>'application_id',
        'client_secret'=>'application_secret',
        'auth_code'=>'auth_code'
    )
);

// and get tokens
$token = $client->authenticate(true);

// $token is an object with access_token, refresh_token, expires_in
```

The best is to store gathered tokens into the database - *access_token* is required every time an application gets access to the shop.

1.5.3 Refreshing the token

In case the token gets expired (look at: `expires_in`) or in case it's invalidated, it's possible to refresh it:

```
$client = Client::factory(
    Client::ADAPTER_OAUTH,
    array(
        'entrypoint'=>'https://shop.url',
        'client_id'=>'application_id',
        'client_secret'=>'application_secret',
        'refresh_token'=>'refresh_token'
    )
);

// and get tokens
$token = $client->refreshTokens();

// $token is an object with access_token, refresh_token, expires_in
```

This object has equal information to the example above.

1.5.4 Performing REST-ful request

With a valid token, it's possible to perform request to the shop according to the API documentation:

```
$client = Client::factory(
    Client::ADAPTER_OAUTH,
    array(
        'entrypoint'=>'http://myshop.example.com',
        'client_id'=>'application_id',
        'client_secret'=>'application_secret'
    )
);

$client->setAccessToken('SHOP TOKEN');

// getting collection/object
$product = new \DreamCommerce\ShopAppstoreLib\Resource\Product($client);
$list = $product->get();

foreach($list as $item){
    //...
}

// object update
$product->put(ID, array(...));

// create object
$product->post(array(...));

// delete object
$product->delete(ID);
```

CHAPTER 2

Classes list

2.1 Client

class `DreamCommerce\ShopAppstoreLib\`**`Client`**

A client library allowing to perform REST-ful requests.

This class implements *`ClientInterface`*. .

2.1.1 exceptions

BasicAuthException

class `DreamCommerce\ShopAppstoreLib\Client\Exception\`**`BasicAuthException`**

An exception raised upon *`DreamCommerce\ShopAppstoreLib\Client\BasicAuth`* library error.

See: `DreamCommerce\ShopAppstoreLib\Exception\Exception`

Exception

class `DreamCommerce\ShopAppstoreLib\Client\Exception\`**`Exception`**

An exception raised upon *`DreamCommerce\ShopAppstoreLib\Client`* library error.

constants

`API_ERROR` an server API error occured - check error message

`ENTRYPOINT_URL_INVALID` invalid shop URL

`METHOD_NOT_SUPPORTED` tried to perform an unsupported method (other than `GET/POST/PUT/DELETE`)

`UNKNOWN_ERROR` unknown error

`PARAMETER_NOT_SPECIFIED` required parameter has not been specified

OAuthException

class DreamCommerce\ShopAppstoreLib\Client\Exception**OAuthException**

An exception raised upon *DreamCommerce\ShopAppstoreLib\Client\OAuth* library error.

See: DreamCommerce\ShopAppstoreLib\Exception\Exception

2.1.2 adapters

BasicAuth

class DreamCommerce\ShopAppstoreLib\Client**BasicAuth**

A client library allowing to perform REST-ful requests using Basic authentication method.

This class implements *DreamCommerce\ShopAppstoreLib\ClientInterface*.

constants

HTTP_ERROR_AUTH_FAILURE Authentication failure

HTTP_ERROR_AUTH_IP_NOT_ALLOWED Failure due to invalid IP being used

HTTP_ERROR_AUTH_WEBAPI_ACCESS_DENIED Missing WebAPI credentials

methods

DreamCommerce\ShopAppstoreLib\Client\BasicAuth::**__construct**(*$options* = [ ])
constructor

Parameters

- **$options** (*array*) – object instantiation options

Throws ClientBasicAuthException

Available options keys:

username auth user name

password auth password

DreamCommerce\ShopAppstoreLib\Client\BasicAuth::**getUsername**()
Get already set user name.

DreamCommerce\ShopAppstoreLib\Client\BasicAuth::**setUsername**(*$username*)
Set user name for request.

Parameters

- **$username** (*string*) – user name

DreamCommerce\ShopAppstoreLib\Client\BasicAuth::**getPassword**()
Get already set password.

DreamCommerce\ShopAppstoreLib\Client\BasicAuth::**setPassword**(*$password*)
Set password for request.

Parameters

- **$password** (*string*) – password

Bearer

class DreamCommerce\ShopAppstoreLib\Client**Bearer**

An abstract client class for most of REST operations.

This class implements *DreamCommerce\ShopAppstoreLib\ClientInterface*.

constants

HTTP_ERROR_INVALID_REQUEST invalid HTTP request

HTTP_ERROR_UNAUTHORIZED_CLIENT connection lacks from authorization data

HTTP_ERROR_ACCESS_DENIED access for specified client is denied

HTTP_ERROR_UNSUPPORTED_RESPONSE_TYPE response type is unsupported

HTTP_ERROR_UNSUPPORTED_GRANT_TYPE grant is unsupported

HTTP_ERROR_INVALID_SCOPE scope is invalid

HTTP_ERROR_INVALID_GRANT grant is invalid

HTTP_ERROR_INSUFFICIENT_SCOPE current token has insufficient permissions

HTTP_ERROR_REDIRECT_URI_MISMATCH problem with redirection

HTTP_ERROR_SERVER_ERROR internal server error occurred

HTTP_ERROR_TEMPORARILY_UNAVAILABLE server returned 5xx error

methods

DreamCommerce\ShopAppstoreLib\Client\Bearer::**__construct**(*$options* = [ ])
constructor

Parameters

- **$options** (*array*) – object instantiation options

Throws ClientException

Available options keys:

entrypoint shop URL

DreamCommerce\ShopAppstoreLib\Client\Bearer::**getAccessToken**()
Get already set token.

Return type string

DreamCommerce\ShopAppstoreLib\Client\Bearer::**setAccessToken**(*$token*)
Set token.

Parameters

- **$token** (*string*) – token

DreamCommerce\ShopAppstoreLib\Client\Bearer::**getExpiresIn**()
Get token expiration.

Return type string

DreamCommerce\ShopAppstoreLib\Client\Bearer::**setExpiresIn**(*$expiresIn*)
Set token expiration

Parameters

- **$expiresIn** (*integer*) – expiration

OAuth

class DreamCommerce\ShopAppstoreLib\Client**OAuth**

An adapter for OAuth requests.

This class implements *DreamCommerce\ShopAppstoreLib\ClientInterface*.

methods

DreamCommerce\ShopAppstoreLib\Client\OAuth::**__construct**(*$options* = [ ])
constructor

Parameters

- **$options** (*array*) – object instantiation options

Throws ClientOAuthException

Available options keys:

entrypoint shop URL

client_id application identifier

client_secret application secret

auth_code authentication code used for access token exchange

refresh_token refresh token used for access token update

DreamCommerce\ShopAppstoreLib\Client\OAuth::**refreshTokens**()
Refreshes access tokens. (implies setting refresh_token in *__construct* or *setRefreshToken*)

Return type array

Returns new tokens

DreamCommerce\ShopAppstoreLib\Client\OAuth::**getClientId**()
Gets supplied application identifier.

Return type stringlnull

Returns identifier

DreamCommerce\ShopAppstoreLib\Client\OAuth::**setClientId**(*$clientId*)
Sets application identifier.

Parameters

- **$clientId** (*string*) – identifier

Return type OAuth

Returns chain

DreamCommerce\ShopAppstoreLib\Client\OAuth::**getClientSecret**()
Gets supplied application secret.

Return type string|null

Returns secret

DreamCommerce\ShopAppstoreLib\Client\OAuth::**setClientSecret**(*$clientSecret*)
Sets application secret.

Parameters

- **$clientSecret** (*string*) – secret

Return type OAuth

Returns chain

DreamCommerce\ShopAppstoreLib\Client\OAuth::**getAuthCode**()
Gets supplied authentication code.

Return type string

Returns code

Throws ClientException

DreamCommerce\ShopAppstoreLib\Client\OAuth::**setAuthCode**(*$authCode*)
Sets authentication code.

Parameters

- **$authCode** (*string*) – code

Return type OAuth

Returns chain

DreamCommerce\ShopAppstoreLib\Client\OAuth::**getRefreshToken**()
Gets supplied refresh token.

Return type string

Returns token

Throws ClientException

DreamCommerce\ShopAppstoreLib\Client\OAuth::**setRefreshToken**(*$refreshToken*)
Sets refresh token.

Parameters

- **$refreshToken** (*string*) – token

Return type OAuth

Returns chain

DreamCommerce\ShopAppstoreLib\Client\OAuth::**getScopes**()
Gets granted scopes list.

Return type array

Returns scopes

2.1.3 constants

constant DreamCommerce\ShopAppstoreLib\Client::**ADAPTER_OAUTH**
a pointer to the *Client\OAuth*

constant DreamCommerce\ShopAppstoreLib\Client::**ADAPTER_BASIC_AUTH**
a pointer to the *Client\BasicAuth*

2.1.4 static methods

static DreamCommerce\ShopAppstoreLib\Client::**factory**(*$adapter*, *$options* =[ ])
Creates client instance.

Parameters

- **$adapter** (*string*) – adapter name
- **$options** (*array*) – creation options

Return type ClientInterface

Returns client

Globally, $options accepts following keys:

entrypoint shop URL

adapterNamespace a namespace used to find $adapter class

static DreamCommerce\ShopAppstoreLib\Client::**getDefaultAdapter**
Gets default defined adapter

Return type ClientInterface|null

Returns client

static DreamCommerce\ShopAppstoreLib\Client::**setDefaultAdapter**(*ClientInterface $adapter*)
Sets default defined adapter

Parameters

- **$adapter** (*ClientInterface*) – adapter handle

Return type void

2.1.5 methods

DreamCommerce\ShopAppstoreLib\Client::**__construct**(*$entrypoint*, *$clientId*, *$clientSecret*)
constructor

Parameters

- **$exception** (*string*) – in case of webapi/rest is not a part of URL, it will be automatically appended
- **$clientId** (*string*) – string application ID
- **$clientSecret** (*string*) – application secret code (generated upon App Store application registration)

Calling a constructor is adequate to perform this call:

```
    self::factory(self::ADAPTER_OAUTH, array(
        'entrypoint' => $entrypoint,
        'client_id' => $clientId,
        'client_secret' => $clientSecret
    )
```

DreamCommerce\ShopAppstoreLib\Client::**getToken**(*$authCode*)
Gets token using the AuthCode.

Parameters

- **$authCode** (*string*) – authorization code, passed during installation

Return type array

Returns an array with tokens

DreamCommerce\ShopAppstoreLib\Client::**refreshToken**(*$refreshToken*)
Refreshes token

Parameters

- **$refreshToken** (*string*) – refresh token, passed during token getting/exchange

Return type array

Returns an array with new tokens

DreamCommerce\ShopAppstoreLib\Client::**setAccessToken**(*$token*)
Sets an access token for current script execution. Called automatically upon exchange/refreshing of token.

Parameters

- **$token** (*string*) – token

DreamCommerce\ShopAppstoreLib\Client::**setAdapter**(*ClientInterface $adapter*)
Sets adapter on this client.

Parameters

- **$adapter** (*ClientInterface*) – adapter

Return type Client

Returns chain

2.2 ClientInterface

interface DreamCommerce\ShopAppstoreLib**ClientInterface**

Interface specifying methods for client modules.

2.2.1 methods

DreamCommerce\ShopAppstoreLib\ClientInterface::**authenticate**(*$force = false*)
Performs authentication based on AuthCode set earlier.

Parameters

- **$force** (*boolean*) – force getting tokens ignoring already set

Return type stdClass

Returns tokens

A sample structure may look like:

```
array(
    'access_token'=>'xxxxxx',
    'expires_in'=>'3600',
    'token_type'=>'bearer'
)
```

DreamCommerce\ShopAppstoreLib\ClientInterface::**request**(*Resource $res*, *$method*, *$objectPath = null*, *$data =[ ]*, *$query =[ ]*)

Performs REST request

Parameters

- **$res** (*Resource*) – resource to perform request against
- **$method** (*string*) – HTTP method name
- **$objectPath** (*null|array|integer*) – URL path of resource
- **$data** (*array*) – payload
- **$query** (*array*) – query string

Return type mixed

DreamCommerce\ShopAppstoreLib\ClientInterface::**setHttpClient**(*HttpInterface $httpClient*)

Set HttpClient for client.

Parameters

- **$httpClient** (*HttpInterface*) – client

DreamCommerce\ShopAppstoreLib\ClientInterface::**getHttpClient**()

Get HttpClient.

Return type HttpInterface|null

DreamCommerce\ShopAppstoreLib\ClientInterface::**getLocale**()

Get current locale.

DreamCommerce\ShopAppstoreLib\ClientInterface::**setLocale**(*$locale*)

Set messages language based on `$locale`.

Parameters

- **$locale** (*string*) – locale to set

DreamCommerce\ShopAppstoreLib\ClientInterface::**getLogger**()

Get bound `LoggerInterface`` instance.

Return type LoggerInterface|null

DreamCommerce\ShopAppstoreLib\ClientInterface::**setLogger**(*LoggerInterface $logger*)

Set `LoggerInterface`` for this client.

Parameters

- **$logger** (*LoggerInterface*) – instance

DreamCommerce\ShopAppstoreLib\ClientInterface::**setOnTokenInvalidHandler**(*$callback = null*)

Set handler called upon invalid token exception detected.

Parameters

- **$callback** (*Callable|null*) – callback

2.3 Exception

class DreamCommerce\\ShopAppstoreLib\\Exception**Exception**

2.3.1 derived classes

HandlerException

class DreamCommerce\ShopAppstoreLib\Exception**HandlerException**

An exception raised upon billing system responder error.

constants

ACTION_HANDLER_NOT_EXISTS action handler doesn't exist

ACTION_NOT_EXISTS tried to handle non-existing action

ACTION_NOT_SPECIFIED an action to handle has not been specified

CLIENT_INITIALIZATION_FAILED an error occurred upon client class initialization - probably invalid data/tokens provided

HASH_FAILED packet checksum is invalid

INCORRECT_HANDLER_SPECIFIED handler for action is invalid

PAYLOAD_EMPTY server returned a response with no data

HttpException

class DreamCommerce\ShopAppstoreLib\Exception**HttpException**

An exception raised upon problem in communication layer (HTTP protocol error).

constants

LIMIT_TOO_LOW specified request limit is too low

METHOD_NOT_SUPPORTED tried to perform an unsupported HTTP method

QUOTA_EXCEEDED requests quota exceeded

REQUEST_FAILED a request failed (eg. I/O)

MALFORMED_RESULT cannot parse server response

methods

__construct([$message = ''[, $code = 0[, $previous = null[, $method = null[, $url = null[,
Method instantiates exception.

Parameters

- **$message** (*string*) – exception message
- **$code** (*integer*) – error code
- **$previous** (*Exception|null*) – a handle to the previous exception
- **$method** (*string*) – HTTP method name
- **$url** (*string*) – URL to entrypoint that caused failure
- **$headers** (*array*) – an array with response headers
- **$query** (*array*) – query string parameters
- **$body** (*array*) – body parameters
- **$response** (*mixed*) – server response
- **$responseHeaders** (*array*) – server response headers

DreamCommerce\ShopAppstoreLib\Exception\HttpException::**getHeaders**()
Returns an array with headers returned upon exception raising.

Return type array

Returns headers

DreamCommerce\ShopAppstoreLib\Exception\HttpException::**getResponse**()
Returns a raw server response that caused an exception.

Return type string

Returns response

DreamCommerce\ShopAppstoreLib\Exception\HttpException::**getMethod**()
Returns HTTP method name.

Return type string

Returns method name

DreamCommerce\ShopAppstoreLib\Exception\HttpException::**getUrl**()
Returns URL.

Return type string

Returns URL

DreamCommerce\ShopAppstoreLib\Exception\HttpException::**getBody**()
Returns body used for request.

Return type mixed

Returns body

DreamCommerce\ShopAppstoreLib\Exception\HttpException::**getQuery**()
Returns query string parameters.

Return type array

Returns query string

DreamCommerce\ShopAppstoreLib\Exception\HttpException::**getResponseHeaders**()
Get returned server response headers.

Return type array

Returns response headers

DreamCommerce\ShopAppstoreLib\Exception\HttpException::**__toString**()
Serializes exception to loggable form.

Return type array

Returns string-serialized exception data

2.4 Handler

class DreamCommerce\ShopAppstoreLib**Handler**

It's an object for handling automatic messages from AppStore. It's event-based which allows to easily bind many handlers to the particular event (eg. for installation handler, it's possible to bind database storing, cache purging, etc).

See *DreamCommerce\ShopAppstoreLib\HandlerInterface*.

2.4.1 methods

DreamCommerce\ShopAppstoreLib\Handler::**__construct**(*$entrypoint*, *$clientId*, *$secret*, *$appStoreSecret*)
Class constructor

Parameters

- **$entrypoint** (*string*) – shop URL - in case of webapi/rest is not a part of URL, it will be automatically appended
- **$clientId** (*string*) – application ID
- **$secret** (*string*) – API key
- **$appStoreSecret** (*string*) – secret code

2.5 HandlerInterface

interface DreamCommerce\ShopAppstoreLib**HandlerInterface**

Interface specifying methods for handler modules.

2.5.1 constants

EVENT_INSTALL install message event name

EVENT_UNINSTALL uninstall message event name

EVENT_BILLING_INSTALL application paid event name

EVENT_BILLING_SUBSCRIPTION application subscription paid event name

EVENT_UPGRADE application upgrade event name

2.5.2 methods

DreamCommerce\ShopAppstoreLib\HandlerInterface::**actionExists**(*$action*)
Checks whether handled action is supported by library

Parameters

- **$action** (*string*) – action name

Return type boolean

Throws DreamCommerceHandlerException

DreamCommerce\ShopAppstoreLib\HandlerInterface::**verifyPayload**(*$payload*)
Verifies provided payload

Parameters

- **$payload** (*array*) – payload to verify

Return type boolean

Throws DreamCommerceHandlerException

DreamCommerce\ShopAppstoreLib\HandlerInterface::**dispatch**(*$requestBody = null*)
Dispatch request to desired action listener

Parameters

- **$requestBody** (*array*) – request body; if null - use $_POST

Return type boolean

Throws DreamCommerceHandlerException

DreamCommerce\ShopAppstoreLib\HandlerInterface::**subscribe**(*$event*, *$handler*)
Subscribes a handler to the chosen event.

Parameters

- **$event** (*string*) – event type for handling by handler
- **$handler** (*Callable*) – a handler to call when subscribed event is fired. $handler will be called with one argument of event params. If handler returns false value, event propagation is stopped.

Available $event values:

- install - an application is being installed in shop
- uninstall - an application is being uninstalled
- billing_install - installation payment
- billing_subscription - subscription payment

DreamCommerce\ShopAppstoreLib\HandlerInterface::**unsubscribe**(*$event*, *$handler = null*)
Unsubscribes from event handling.

Parameters

- **$event** (*string*) – event type for handling by handler
- **$handler** (*null|Callable*) – if is null, all handlers are unbound. Specifying particular handler, leaves alone all except chosen.

Available $event values:

- `install` - an application is being installed in shop
- `uninstall` - an application is being uninstalled
- `billing_install` - installation payment
- `billing_subscription` - subscription payment

`DreamCommerce\ShopAppstoreLib\HandlerInterface::`**`setClient`**(*ClientInterface $client*)
Sets client on this handler.

Parameters

- **`$client`** (*`ClientInterface`*) – client

Return type Client

Returns chain

`DreamCommerce\ShopAppstoreLib\HandlerInterface::`**`getClient`**()
Gets adapter bound to the handler.

Return type ClientInterface

Returns client

`DreamCommerce\ShopAppstoreLib\HandlerInterface::`**`getLogger`**()
Get bound `LoggerInterface` instance.

Return type LoggerInterface|null

`DreamCommerce\ShopAppstoreLib\HandlerInterface::`**`setLogger`**(*LoggerInterface $logger*)
Set `LoggerInterface` for this client.

Parameters

- **`$logger`** (*`LoggerInterface`*) – instance

2.6 Http

class `DreamCommerce\ShopAppstoreLib\`**`Http`**

A class performing HTTP requests.

This class implements *`HttpInterface`*.

2.6.1 static methods

static `DreamCommerce\ShopAppstoreLib\Http::`**`instance`**
Returns a class instance

Returns class instance

Return type Http

static `DreamCommerce\ShopAppstoreLib\Http::`**`setRetryLimit`**(*$num*)
Sets retries count if requests quota is exceeded.

Parameters

- **`$num`** (*`integer`*) – retries limit

2.6.2 methods

DreamCommerce\ShopAppstoreLib\Http::**parseHeaders**(*$src*)

Parse `$http_response_header` to more readable form.

Parameters

- **$src** (*array*) – source headers data

Return type array

Returns parsed headers

Returned array looks like:

```
array(
    'Code'=>404,
    'Status'=>'Not Found'
    'X-Header1'=>'value',
    'X-Header2'=>'value'
)
```

DreamCommerce\ShopAppstoreLib\Http::**setSkipSsl**(*$status*)

Tell PHP to skip SSL certificates validation.

Parameters

- **$status** (*boolean*) – skip?

Return type void

2.7 HttpInterface

interface DreamCommerce\ShopAppstoreLib**HttpInterface**

Interface specifying methods for HTTP communication layer.

Each of implemented methods returns following data set:

```
array(
    'headers' => (
        'Content-Type' => 'application/json'
    ),
    'data' => <\ArrayObject|string>
)
```

2.7.1 methods

DreamCommerce\ShopAppstoreLib\HttpInterface::**getLogger**()

Get bound `LoggerInterface`` instance.

Return type LoggerInterface|null

DreamCommerce\ShopAppstoreLib\HttpInterface::**setLogger**(*LoggerInterface $logger*)

Set `LoggerInterface`` for this client.

Parameters

- **$logger** (*LoggerInterface*) – instance

DreamCommerce\ShopAppstoreLib\HttpInterface::**delete**(*$url*[, *$query = array()*[, *$headers = array()*]])

Performs HTTP DELETE.

Parameters

- **$url** (*string*) – URL
- **$query** (*array*) – query parameters (URL query string, after question mark)
- **$headers** (*array*) – additional headers to sent within request

Return type array

Returns see: *structure*

DreamCommerce\ShopAppstoreLib\HttpInterface::**head**(*$url*[, *$query = array()*[, *$headers = array()*]])

Performs HTTP HEAD.

Parameters

- **$url** (*string*) – URL
- **$query** (*array*) – query parameters (URL query string, after question mark)
- **$headers** (*array*) – additional headers to sent within request

Return type array

Returns see: *structure*

DreamCommerce\ShopAppstoreLib\HttpInterface::**get**(*$url*[, *$query = array()*[, *$headers = array()*]])

Performs HTTP GET.

Parameters

- **$url** (*string*) – URL
- **$query** (*array*) – query parameters (URL query string, after question mark)
- **$headers** (*array*) – additional headers to sent within request

Return type array

Returns see: *structure*

DreamCommerce\ShopAppstoreLib\HttpInterface::**post**(*$url*[, *$body = array()*[, *$query = array()*[, *$headers = array()*]]])

Performs HTTP POST.

Parameters

- **$url** (*string*) – URL
- **$body** (*string*) – request body
- **$query** (*array*) – query parameters (URL query string, after question mark)
- **$headers** (*array*) – additional headers to sent within request

Return type mixed

Returns see: *structure*

DreamCommerce\ShopAppstoreLib\HttpInterface::**put**(*$url*[, *$body = array()*[, *$query = array()*[, *$headers = array()*]]])

Performs HTTP PUT.

Parameters

- **$url** (*string*) – URL
- **$body** (*string*) – request body
- **$query** (*array*) – query parameters (URL query string, after question mark)
- **$headers** (*array*) – additional headers to sent within request

Return type mixed

Returns see: *structure*

`DreamCommerce\ShopAppstoreLib\HttpInterface::getLogger()`
Get bound `LoggerInterface` instance.

Return type LoggerInterface|null

`DreamCommerce\ShopAppstoreLib\HttpInterface::setLogger(LoggerInterface $logger)`
Set `LoggerInterface` for this client.

Parameters

- **$logger** (*LoggerInterface*) – instance

2.8 Logger

class `DreamCommerce\ShopAppstoreLib\Logger`

A class performing simple messages logging.

This class implements PSR-3 Logger Interface.

2.8.1 methods

`DreamCommerce\ShopAppstoreLib\Logger::log($level, $message[, $context = [ ]])`
Logs message to defined stream.

Parameters

- **$level** (*string*) – priority
- **$message** (*string*) – log message
- **$context** (*array*) – logging context

The target stream can be set by defining `DREAMCOMMERCE_LOG_FILE`

You can define the constant in your source code:

```
define('DREAMCOMMERCE_LOG_FILE', "php://stdout");
```

By default, all messages are added with debug priority. All those messages are by default filtered out, due to disabled debug mode.

Debugging may be enabled by defining `DREAMCOMMERCE_DEBUG` constant and setting its value to `true`.

You can define the constant in your source code:

```
define('DREAMCOMMERCE_DEBUG', true);
```

2.9 Resource

class DreamCommerce\ShopAppstoreLib**Resource**

Represents particular resource.

2.9.1 resources

AboutPage

class DreamCommerce\ShopAppstoreLib\Resource**AboutPage**

Check: Resource.

AdditionalField

class DreamCommerce\ShopAppstoreLib\Resource**AdditionalField**

Check: Resource.

constants

FIELD_TYPE_TEXT text input

FIELD_TYPE_CHECKBOX checkbox field

FIELD_TYPE_SELECT select (drop down)

FIELD_TYPE_FILE file input

FIELD_TYPE_HIDDEN hidden field

LOCATE_USER place field in user context

LOCATE_USER_ACCOUNT place field in user account panel

LOCATE_USER_REGISTRATION place field in user registration

LOCATE_ORDER place field in order

LOCATE_ORDER_WITH_REGISTRATION place field when user is being registered upon order

LOCATE_ORDER_WITHOUT_REGISTRATION place field when user is not being registered upon order

LOCATE_ORDER_LOGGED_ON_USER place field when user is logged in upon order

LOCATE_CONTACT_FORM place field in contact form

ApplicationLock

class DreamCommerce\ShopAppstoreLib\Resource**ApplicationLock**

Check: Resource.

ApplicationConfig

class `DreamCommerce\ShopAppstoreLib\Resource\`**ApplicationConfig**

Check: Resource.

constants

CONFIG_BLOG_COMMENTS_ENABLE Enable blog comments

CONFIG_BLOG_COMMENTS_FOR_USERS Only registered users are allowed to post blog comments

CONFIG_BLOG_COMMENTS_MODERATION Is blog comments moderation enabled - boolean

CONFIG_BLOG_DOWNLOAD_FOR_USERS Allow to download blog attached files - boolean

CONFIG_BLOG_ITEMS_PER_PAGE Blog items per page count

CONFIG_COMMENT_ENABLE Enable product comments - boolean

CONFIG_COMMENT_FOR_USERS Only registered users are allowed to post comments - boolean

CONFIG_COMMENT_MODERATION Is comments moderation enabled - boolean

CONFIG_DEFAULT_CURRENCY_ID Default currency identifier - integer

CONFIG_DEFAULT_CURRENCY_NAME Default currency name - ISO 4217 (“PLN”) - string

CONFIG_DEFAULT_LANGUAGE_ID Default shop language identifier - integer

CONFIG_DEFAULT_LANGUAGE_NAME Default shop language name - language_REGION format (eg. “pl_PL”) - string

CONFIG_DIGITAL_PRODUCT_LINK_EXPIRATION_TIME Product link expiration time (days) - integer

CONFIG_DIGITAL_PRODUCT_NUMBER_OF_DOWNLOADS Maximum downloads number per user - integer

CONFIG_DIGITAL_PRODUCT_UNLOCKING_STATUS_ID Status identifier which sends email with files - integer

CONFIG_LOCALE_DEFAULT_WEIGHT Shop weight unit:

- KILOGRAM
- GRAM
- LBS

CONFIG_LOYALTY_ENABLE Is loyalty program enabled - boolean

CONFIG_PRODUCT_DEFAULTS_ACTIVE Is product active in default - boolean

CONFIG_PRODUCT_DEFAULTS_AVAILABILITY_ID Default availability identifier - integer

CONFIG_PRODUCT_DEFAULTS_CODE Default product code generating method, available values:

- 1 - no method
- 2 - following ID
- 3 - random

CONFIG_PRODUCT_DEFAULTS_DELIVERY_ID Default product delivery identifier - integer

CONFIG_PRODUCT_DEFAULTS_ORDER Default ordering factor value - integer

CONFIG_PRODUCT_DEFAULTS_STOCK Default stock value - integer

CONFIG_PRODUCT_DEFAULTS_TAX_ID Default tax value identifier - integer

CONFIG_PRODUCT_DEFAULTS_UNIT_ID Default measurement unit identifier - integer

CONFIG_PRODUCT_DEFAULTS_WARN_LEVEL Default stock value warning level - integer

CONFIG_PRODUCT_DEFAULTS_WEIGHT Default product weight - float

CONFIG_PRODUCT_NOTIFIES_ENABLE Notify on product availabilities - boolean

CONFIG_PRODUCT_SEARCH_ALL_TOKENS Only for product name search - require exact phrase

CONFIG_PRODUCT_SEARCH_CODE Only for product details search - include product code

CONFIG_PRODUCT_SEARCH_DESCRIPTION Only for product details search - include product description - boolean

CONFIG_PRODUCT_SEARCH_SHORT_DESCRIPTION Only for product details search - include product short description

CONFIG_PRODUCT_SEARCH_TYPE Product search mode:

- 1 - only in product name
- 2 - in product name and other details

CONFIG_REGISTRATION_CONFIRM Require registration confirmation - boolean

CONFIG_REGISTRATION_ENABLE Enable user registration - boolean

CONFIG_REGISTRATION_LOGIN_TO_SEE_PRICE Only signed in users see price - boolean

CONFIG_REGISTRATION_REQUIRE_ADDRESS New users address requirements:

- 0 - only email address and password
- 1 - full address details

CONFIG_SHIPPING_ADD_PAYMENT_COST_TO_FREE_SHIPPING Force payment price addition even if free shipping is present - boolean

CONFIG_SHIPPING_VOLUMETRIC_WEIGHT_ENABLE Enable volumetric weight - boolean

CONFIG_SHOPPING_ALLOW_OVERSELLING Allow to sell more products than stock value - boolean

CONFIG_SHOPPING_ALLOW_PRODUCT_DIFFERENT_CURRENCY Allow to set currency per product - boolean

CONFIG_SHOPPING_ALLOW_TO_BU_NOT_REG Allow buying without registration - boolean

CONFIG_SHOPPING_BASKET_ADDING Actions performed upon product adding, available values:

- 1 - refresh page and do not redirect to the basket
- 2 - refresh page and perform redirection to the basket
- 3 - do not refresh page, show confirmation message

CONFIG_SHOPPING_BESTSELLER_ALGORITHM Bestseller calculation algorithm:

- 1 - most orders count
- 2 - most orders amount

CONFIG_SHOPPING_BESTSELLER_DAYS Only for automatic mode - days count when product is marked as bestseller, available values:

- 7 - last 7 days
- 30 - last 30 days
- 90 - last 90 days

- 0 - lifetime

CONFIG_SHOPPING_BESTSELLER_MODE Marking as "bestseller" mode:

- 0 - manual
- 1 - automatic, based on users orders

CONFIG_SHOPPING_CONFIRM_ORDER Require order confirmation - boolean

CONFIG_SHOPPING_MIN_ORDER_VALUE Minimal order value - float

CONFIG_SHOPPING_MIN_PROD_QUANTITY Minimal product quantity - float

CONFIG_SHOPPING_NEWPRODUCTS_DAYS Automatic mode - number of days after product creation it will be marked as "new" - integer

CONFIG_SHOPPING_NEWPRODUCTS_MODE Products marking as "new" mode, available values:

- 0 - manual
- 1 - automatic, based on product creation date

CONFIG_SHOPPING_OFF Disable shopping - boolean

CONFIG_SHOPPING_ORDER_VIA_TOKEN Share order via link - boolean

CONFIG_SHOPPING_PARCEL_CREATE_STATUS_ID Order status after parcel is created - integer|null

CONFIG_SHOPPING_PARCEL_SEND_STATUS_ID Order status after parcel is sent - integer|null

CONFIG_SHOPPING_PRICE_COMPARISON_FIELD Field which products are identified by - for price comparison websites, available values:

- code - product code
- additional_isbn - ISBN code
- additional_kgo - KGO price
- additional_bloz7 - BLOZ7 code
- additional_bloz12 - BLOZ12 code

CONFIG_SHOPPING_PRICE_LEVELS Defined price levels (1-3) - integer

CONFIG_SHOPPING_PRODUCTS_ALLOW_ZERO Allow to buy zero-priced products - boolean

CONFIG_SHOPPING_SAVE_BASKET Save basket contents - boolean

CONFIG_SHOPPING_SHIPPING_EXTRA_STEP Show shipping and payment, available values:

- 0 - show in basket
- 1 - show as separated step

CONFIG_SHOPPING_UPDATE_STOCK_ON_BUY Update stock values on buy - boolean

CONFIG_SHOP_ADDRESS_1 Shop address line 1 - string

CONFIG_SHOP_ADDRESS_2 Shop address line 2 - string

CONFIG_SHOP_CITY Shop city - string

CONFIG_SHOP_COMPANY_NAME Company name - string

CONFIG_SHOP_COUNTRY Shop country code - string

CONFIG_SHOP_EMAIL Shop mail e-mail address - string

CONFIG_SHOP_FULL_ADDRESS Shop full address - string

CONFIG_SHOP_NAME Full shop name - string

CONFIG_SHOP_OFF Is shop disabled - boolean

CONFIG_SHOP_PHONE Shop phone number - string

CONFIG_SHOP_PROVINCE Shop province - string

CONFIG_SHOP_REGON Company identification number - integer

CONFIG_SHOP_TAX_ID Tax identifier - integer

CONFIG_SHOP_TRADE Shop trade - string

CONFIG_SHOP_TRADE_CODE Shop trade code, available values:

- children
- books_and_multimedia
- clothes
- computers
- delicatessen
- gifts_and_accessories
- health_and_beauty
- hobby
- home_and_garden
- automotive
- consumer_electronics
- sport_and_travel
- other

CONFIG_SHOP_URL Shop URL - string

CONFIG_SHOP_ZIP_CODE Shop postcode - string

ApplicationVersion

class `DreamCommerce\ShopAppstoreLib\Resource\`**ApplicationVersion**

Check: Resource.

AttributeGroup

class `DreamCommerce\ShopAppstoreLib\Resource\`**AttributeGroup**

Check: Resource.

Attribute

class `DreamCommerce\ShopAppstoreLib\Resource\`**Attribute**

Check: Resource.

constants

TYPE_TEXT text field

TYPE_CHECKBOX checkbox field

TYPE_SELECT select field

Auction

class DreamCommerce\ShopAppstoreLib\Resource**Auction**

Check: Resource.

constants

SALES_FORMAT_BIDDING auction is bid-based

SALES_FORMAT_IMMEDIATE "buy now"

SALES_FORMAT_SHOP treat auction just like a shop

AuctionHouse

class DreamCommerce\ShopAppstoreLib\Resource**AuctionHouse**

Check: Resource.

AuctionOrder

class DreamCommerce\ShopAppstoreLib\Resource**AuctionOrder**

Check: Resource.

Availability

class DreamCommerce\ShopAppstoreLib\Resource**Availability**

Check: Resource.

Category

class DreamCommerce\ShopAppstoreLib\Resource**Category**

Check: Resource.

CategoriesTree

class DreamCommerce\ShopAppstoreLib\Resource**CategoriesTree**

Check: Resource.

Currency

class DreamCommerce\ShopAppstoreLib\Resource**Currency**

Check: Resource.

DashboardActivity

class DreamCommerce\ShopAppstoreLib\Resource**DashboardActivity**

Check: Resource.

DashboardStat

class DreamCommerce\ShopAppstoreLib\Resource**DashboardStat**

Check: Resource.

Delivery

class DreamCommerce\ShopAppstoreLib\Resource**Delivery**

Check: Resource.

Gauge

class DreamCommerce\ShopAppstoreLib\Resource**Gauge**

Check: Resource.

GeolocationCountry

class DreamCommerce\ShopAppstoreLib\Resource**GeolocationCountry**

Check: Resource.

GeolocationRegion

class DreamCommerce\ShopAppstoreLib\Resource**GeolocationRegion**

Check: Resource.

GeolocationSubregion

class DreamCommerce\ShopAppstoreLib\Resource**GeolocationSubregion**

Check: Resource.

Language

class DreamCommerce\ShopAppstoreLib\Resource**Language**

Check: Resource.

Metafield

class DreamCommerce\ShopAppstoreLib\Resource**Metafield**

Check: Resource.

constants

TYPE_INT type of integer

TYPE_FLOAT type of float

TYPE_STRING type of string

TYPE_BLOB type of binary data

MetafieldValue

class DreamCommerce\ShopAppstoreLib\Resource**MetafieldValue**

Check: Resource.

ObjectMtime

class DreamCommerce\ShopAppstoreLib\Resource**ObjectMtime**

Check: Resource.

OptionGroup

class DreamCommerce\ShopAppstoreLib\Resource**OptionGroup**

Check: Resource.

Option

class DreamCommerce\ShopAppstoreLib\Resource**Option**

Check: Resource.

constants

HTTP_ERROR_OPTION_CAN_NOT_MODIFY_REQUIRE it's not possible to change required flag for option with warehouse support

HTTP_ERROR_OPTION_CAN_NOT_MODIFY_TYPE it's not possible to change type of an existing option

TYPE_FILE option type file input

TYPE_TEXT option type text

TYPE_RADIO option type radio option

TYPE_SELECT option type select (drop down)

TYPE_CHECKBOX option type checkbox

TYPE_COLOR option type color

PRICE_TYPE_DECREASE option value decreases price

PRICE_TYPE_KEEP option value is unchanged

PRICE_TYPE_INCREASE option value increases price

PRICE_PERCENT option value is modified by percent

PRICE_AMOUNT option value is modified by value

OptionValue

class `DreamCommerce\ShopAppstoreLib\Resource\`**OptionValue**

Check: Resource.

constants

HTTP_ERROR_OPTION_CHILDREN_NOT_SUPPORTED option type doesn't support values management

PRICE_TYPE_DECREASE option value decreases price

PRICE_TYPE_KEEP option value keeps price unchanged

PRICE_TYPE_INCREASE option value increases price

PRICE_PERCENT price change by percent

PRICE_AMOUNT price change by value

OrderProduct

class `DreamCommerce\ShopAppstoreLib\Resource\`**OrderProduct**

Check: Resource.

Order

class `DreamCommerce\ShopAppstoreLib\Resource\`**Order**

Check: Resource.

constants

HTTP_ERROR_ORDER_COMBINED combined order has been detected

ORIGIN_SHOP order comes from shop

ORIGIN_FACEBOOK order comes from Facebook

ORIGIN_MOBILE order comes from mobile

ORIGIN_ALLEGRO order comes from Allegro

ORIGIN_WEBAPI order comes from WebAPI

FIELD_TYPE_TEXT order field type is text

FIELD_TYPE_CHECKBOX order field type is checkbox

FIELD_TYPE_SELECT order field type is select (drop down)

FIELD_SHOW_ORDER place field in order

FIELD_SHOW_REGISTERED place field if user is being registered

FIELD_SHOW_GUEST place field if user is not registered

FIELD_SHOW_SIGNED_IN place field if user is signed in

ADDRESS_TYPE_BILLING address is for billing purposes

ADDRESS_TYPE_DELIVERY address is for delivery purposes

Parcel

class `DreamCommerce\ShopAppstoreLib\Resource\`**Parcel**

Check: Resource.

constants

HTTP_ERROR_PARCEL_CAN_NOT_MODIFY It's not possibly to modify shipped parcel except shipping code

HTTP_ERROR_PARCEL_IS_ALREADY_SENT Parcel has been already sent

ADDRESS_TYPE_BILLING address used for billing purposes

ADDRESS_TYPE_DELIVERY address used for delivery purposes

Payment

class `DreamCommerce\ShopAppstoreLib\Resource\`**Payment**

Check: Resource.

Producer

class `DreamCommerce\ShopAppstoreLib\Resource\`**Producer**

Check: Resource.

Product

class `DreamCommerce\ShopAppstoreLib\Resource\`**Product**

Check: Resource.

ProductFile

class `DreamCommerce\ShopAppstoreLib\Resource\`**ProductFile**

Check: Resource.

ProductImage

class DreamCommerce\ShopAppstoreLib\Resource**ProductImage**

Check: Resource.

ProductStock

class DreamCommerce\ShopAppstoreLib\Resource**ProductStock**

Check: Resource.

constants

PRICE_TYPE_KEEP keep base price

PRICE_TYPE_NEW specify price for stock

PRICE_TYPE_INCREASE increase base price

PRICE_TYPE_DECREASE decrease base price

WEIGHT_TYPE_KEEP keep base weight

WEIGHT_TYPE_NEW specify weight for stock

WEIGHT_TYPE_INCREASE increase base weight

WEIGHT_TYPE_DECREASE decrease base weight

Shipping

class DreamCommerce\ShopAppstoreLib\Resource**Shipping**

Check: Resource.

Status

class DreamCommerce\ShopAppstoreLib\Resource**Status**

Check: Resource.

constants

TYPE_NEW status: new

TYPE_OPENED status: opened

TYPE_CLOSED status: closed

TYPE_UNREALIZED status: not completed

SubscriberGroup

class DreamCommerce\ShopAppstoreLib\Resource**SubscriberGroup**

Check: Resource.

Subscriber

class DreamCommerce\ShopAppstoreLib\Resource**Subscriber**

Check: Resource.

Tax

class DreamCommerce\ShopAppstoreLib\Resource**Tax**

Check: Resource.

Unit

class DreamCommerce\ShopAppstoreLib\Resource**Unit**

Check: Resource.

UserAddress

class DreamCommerce\ShopAppstoreLib\Resource**UserAddress**

Check: Resource.

UserGroup

class DreamCommerce\ShopAppstoreLib\Resource**UserGroup**

Check: Resource.

User

class DreamCommerce\ShopAppstoreLib\Resource**User**

Check: Resource.

constants

ORIGIN_SHOP user created via shop

ORIGIN_FACEBOOK user created via Facebook

ORIGIN_MOBILE user created via mobile

ORIGIN_ALLEGRO user created via Allegro

FIELD_TYPE_TEXT text field

FIELD_TYPE_CHECKBOX checkbox

FIELD_TYPE_SELECT select (drop down)

FIELD_SHOW_USER show field for user

FIELD_SHOW_CLIENT show field for client

FIELD_SHOW_REGISTRATION show field during registration

Webhook

class `DreamCommerce\ShopAppstoreLib\Resource\`**Webhook**

Check: Resource.

constants

FORMAT_JSON webhook data encoded using JSON

FORMAT_XML webhook data encoded using XML

EVENT_ORDER_CREATE webhook bound to order create event

EVENT_ORDER_EDIT webhook bound to order edit event

EVENT_ORDER_PAID webhook bound to order is paid event

EVENT_ORDER_STATUS webhook bound to order status change event

EVENT_ORDER_DELETE webhook bound to order delete event

EVENT_CLIENT_CREATE webhook bound to client create event

EVENT_CLIENT_EDIT webhook bound to client edit event

EVENT_CLIENT_DELETE webhook bound to client delete event

EVENT_PRODUCT_CREATE webhook bound to product create event

EVENT_PRODUCT_EDIT webhook bound to product edit event

EVENT_PRODUCT_DELETE webhook bound to product delete event

EVENT_PARCEL_CREATE webhook bound to parcel create event

EVENT_PARCEL_DISPATCH webhook bound to parcel dispatch event

EVENT_PARCEL_DELETE webhook bound to parcel delete event

Zone

class `DreamCommerce\ShopAppstoreLib\Resource\`**Zone**

Check: Resource.

constants

ZONE_MODE_COUNTRIES zone division by countries

ZONE_MODE_REGIONS zone division by regions

ZONE_MODE_CODES zone supports post codes

resource	description
AboutPage	About page
AdditionalField	Additional field
ApplicationLock	Administrator panel lock
ApplicationConfig	Application config

Continued on next page

Table 2.1 – continued from previous page

resource	description
ApplicationVersion	Application config
AttributeGroup	Attributes group
Attribute	Attribute
Auction	Auction
AuctionHouse	Auction house
AuctionOrder	Auction order
Availability	Product availability
Category	Category
CategoriesTree	Category tree
Currency	Currency
DashboardActivity	Dashboard activity
DashboardStat	Sales stats
Delivery	Delivery
Gauge	Gauge
GeolocationCountry	Geolocation country
GeolocationRegion	Geolocation region
GeolocationSubregion	Geolocation subregion
Language	Language
Metafield	Metafield
MetafieldValue	Metafield Value
ObjectMtime	Modification timestamp of object
OptionGroup	Option group
Option	Variant
OptionValue	Variant value
OrderProduct	Product of order
Order	Order
Parcel	Parcel
Payment	Payment method
Producer	Producer
Product	Product
ProductFile	Product file
ProductImage	Photo of product
ProductStock	Product stock
Shipping	Shipping method
Status	Order status
SubscriberGroup	Subscriber group
Subscriber	Subscriber
Tax	Tax value
Unit	Unit of measurement
UserAddress	Shipping address
UserGroup	User group
User	User
Webhook	Webhook
Zone	Zone

2.9.2 exceptions

ResourceException

class `DreamCommerce\ShopAppstoreLib\Resource\Exception\`**`ResourceException`**

Base exception for resource manipulation.

constants

`MALFORMED_RESPONSE` server response is not parseable

`CLIENT_ERROR` client error library occurred (you can reach client exception using `getPrevious()` method)

`LIMIT_BEYOND_RANGE` a limit of maximum simultaneous connections is incorrectly specified

`FILTERS_NOT_SPECIFIED` empty filters specified

`ORDER_NOT_SUPPORTED` tried to sort data by non-existing/not supported field

`INVALID_PAGE` specified results page is beyond the pages count

CommunicationException

class `DreamCommerce\ShopAppstoreLib\Resource\Exception\`**`CommunicationException`**

An exception raised upon problems with server communication that cannot be handled using more specific exceptions.

Check: ResourceException.

NotFoundException

class `DreamCommerce\ShopAppstoreLib\Resource\Exception\`**`NotFoundException`**

An exception raised if target object does not exist.

Check: ResourceException.

ObjectLockedException

class `DreamCommerce\ShopAppstoreLib\Resource\Exception\`**`ObjectLockedException`**

An exception raised if target object is locked by manipulation in admin panel.

PermissionsException

class `DreamCommerce\ShopAppstoreLib\Resource\Exception\`**`PermissionsException`**

An exception raised if token has insufficient permissions to access specified object.

Check: ResourceException.

ValidationException

class DreamCommerce\ShopAppstoreLib\Resource\Exception**ValidationException**

An exception raised if there's a problem with supplied data validity.

Check: ResourceException.

exception	description
CommunicationException	Server communication problem
NotFoundException	Object not found
ObjectLockedException	Object is locked for editing
PermissionsException	Insufficient permissions for object
ResourceException	Base class for exceptions
ValidationException	Supplied data is invalid

2.9.3 static methods

static DreamCommerce\ShopAppstoreLib\Resource::**factory**(*$client*, *$name*)
instantiates new Resource object

Parameters

- **$client** (*Client*) – handle to the client library instance
- **$name** (*string*) – name of resource

Return type Resource

```
\DreamCommerce\ShopAppstoreLib\Resource::factory($client, "product");
```

2.9.4 methods

DreamCommerce\ShopAppstoreLib\Resource::**delete**([*$id = null*])
Deletes an object by ID.

Parameters

- **$id** (*integer*) – object ID

Return type boolean

DreamCommerce\ShopAppstoreLib\Resource::**filters**(*$filters*)
Filters records (GET only).

Chaining method - returns a handle to an object itself.

Parameters

- **$filters** (*array*) – an array of filters

Return type Resource

Returns chain

DreamCommerce\ShopAppstoreLib\Resource::**get**([...])
Performs GET request.

Parameters

- **..** (*mixed*) – arguments - no argument: returns collection - single argument - an object with specified ID - more arguments - resource dependent

Return type ResourceList|ArrayObject|string

DreamCommerce\ShopAppstoreLib\Resource**::head**([...])
Performs HEAD request. Similar to *Resource::get* but without data.

Parameters

- **..** (*mixed*) – arguments - no argument: returns collection - single argument - an object with specified ID - more arguments - resource dependent

Return type ResourceList|ArrayObject|string

DreamCommerce\ShopAppstoreLib\Resource**::getName**()
Returns a plural name of resource.

Return type string

DreamCommerce\ShopAppstoreLib\Resource**::limit**(*$count*)
Restrict amount of fetched records of collection (GET only).

Chaining method - returns a handle to an object itself.

Parameters

- **$count** (*integer*) – count of records to fetch

Return type Resource

Returns chain

DreamCommerce\ShopAppstoreLib\Resource**::order**(*$expr*)
Sorts records by specified criteria (GET only).

Chaining method - returns a handle to an object itself.

Parameters

- **$expr** (*string*) – sorting expression, eg. `field DESC`, `field ASC` or `+field`, `-field`

Return type Resource

Returns chain

DreamCommerce\ShopAppstoreLib\Resource**::page**(*$page*)
Specifies results page number for fetching (GET only).

Chaining method - returns a handle to an object itself.

Parameters

- **$page** (*integer*) – number of page

Return type Resource

Returns chain

DreamCommerce\ShopAppstoreLib\Resource**::post**([*$data = array()*])
Performs a POST request.

Chaining method - returns a handle to an object itself.

Parameters

- **$data** (*array*) – request body

Return type ArrayObject|string

DreamCommerce\ShopAppstoreLib\Resource::**put**([*$id = null*[, *$data = array()*]])
Performs PUT request.

Chaining method - returns a handle to an object itself.

Parameters

- **$id** (*null|integer*) – ID of modified object
- **$data** (*array*) – request body

Return type ArrayObject|string

DreamCommerce\ShopAppstoreLib\Resource::**reset**()
Resets all pagination, limits and filters on current instance.

2.10 ResourceList

class DreamCommerce**ResourceList**

Represents collection of resources, extends ArrayObject

2.10.1 methods

DreamCommerce\ResourceList::**__construct**(*$array = array(), $flags = ArrayObject::ARRAY_AS_PROPS*)
Creates a new class instance.

Parameters

- **$array** (*array*) – Objects in the collection
- **$flags** (*integer*) – flags for ArrayObject

Return type void

DreamCommerce\ResourceList::**setCount**(*$count*)
set number of all resources on requested list.

Parameters

- **$count** (*integer*) – number of all resources on the list

Return type void

DreamCommerce\ResourceList::**getCount**()
get number of all resources on requested list.

Return type integer

Returns number of objects

DreamCommerce\ResourceList::**setPage**(*$page*)
set current page number.

Parameters

- **$page** (*integer*) – current page number

Return type void

DreamCommerce\ResourceList::**getPage**()
get current page number.

Returns page number

Return type integer

DreamCommerce\ResourceList::**setPageCount**(*$count*)
set total page count.

Parameters

- **$count** (*integer*) – total page count

Return type void

DreamCommerce\ResourceList::**getPageCount**()
get total page count.

Return type integer

Returns pages count

CHAPTER 3

Application template

A template (boilerplate) allows to create an application in quickly way using the least amount of development.

It consists of a library and its sample implementation - billing system and main application.

3.1 Structure

The template allows to use simplified MVC pattern in order to separate particular application components.

Creating a new action step-by-step:

- To create new controller, create the class in `src/Controller/` (i.e. `src/Controller/Books.php`), that inherits from `ControllerAbstract`, and is defined in `Controller` namespace
- Create `myAction` method
- Create a view in directory `view` according to the schema: `controller-name/action-name.php` (i.e. `controller/my.php`)
- In order to pass a variable to the view, use following expression `$this['variable'] = 'value';` in action code. Then, this variable will be accessible like a regular variable within the view.

3.2 How to start

- Download the boilerplate
- *Install SDK*
- *Configure*

3.3 Install SDK

If you're using Composer to manage your packages and SDK, inside the project directory call `composer install`.

Alternatively, it's possible to manually install SDK. Just extract its contents into project directory.

3.4 Configure

In file `src/Config.php`:

```
return array(
    /*
     * Application ID generated by AppStore
     */
    'appId' => '',

    /*
     * Secret generated by AppStore
     */
    'appSecret' => '',

    /*
     * AppStore Secret generated by AppStore
     */
    'appstoreSecret' => '',


    'db' => array(
        /*
         * PDO DSN
         */
        'connection' => 'mysql:host=127.0.0.1;dbname=app',

        /*
         * PDO Database username
         */
        'user' => '',

        /*
         * PDO Database password
         */
        'pass' => ''
    ),

    /*
     * Enable debug mode or not
     */
    'debug' => false,

    /*
     * Path to log file or empty to disable logging
     */
    'logFile' => "logs/application.log",


    /*
     * timezone of the application
     *
     * Value is passed to date_default_timezone_set function
     */
    'timezone' => 'Europe/Warsaw',

    'php' => array(
        /*
```

```
         * This determines whether errors should be printed to the screen as
         * part of the output or if they should be hidden from the user
         *
         * Value is passed to ini_set function
         */
        'display_errors' => 'off'
    )
);
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

d

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Z